FISITA2010-SC-O-14

# HYBRID SYSTEMS CONTROLLER DESIGN METHODOLOGY

[1]Pluska Michal*, [1]Sinclair David
[1]LERO@DCU[i], Dublin City University, School of Computing, Dublin 9, Ireland
michal.pluska2@mail.dcu.ie

KEYWORDS - hybrid automata, control software, design methodology, automotive system, system validation

## ABSTRACT

Around ninety percent of vehicle innovations are driven mainly by electronics. The software implementing control algorithms combines the sensor values and calculates some meaningful actuator signals. On the other hand software in the vehicle can be seen as a part of a hybrid system. The hybrid system is a dynamic system that can have both continuous and discrete dynamic behaviour, its mean a system is described by both a differential equation and a difference equation. Moreover hybrid system has the benefit of combining a larger class of systems within its structure, allowing for more flexibility in modelling dynamics and with success can be used to design automotive systems. In up to date, state of the art research there is no methodology to combine these two different domains. Most of the automotive software methodologies are focus on the development of the system which can be describe as a discreet or on the architecture of software alone. On the other hand up to date hybrid system design is focused only on the physical properties of systems and do not consider cooperation of control software and hardware. The on going research aims to combine these two different domains. Presented methodology extends V-model approach to the system design and allows handle complex systems with discreet and continuous dynamics throughout whole design process. It also can be seen as a extension of AUTOMODE process to handle more complicated systems. The graphical interface based on widely adopted UML and comparable SysML, aims for more understandable and generic usage. The design methodology will be described by presenting an example of whole design process for controller of the vehicle active suspension. The proposed design methodology is focus on design process of control algorithm and more important safety verification of it. The methodology will allow safety and time analysis in easier and more structured way. Moreover it will allow parametric approach to the system in design with possibility to find correct parameters of it work by verification. The underling verification process is based on the hybrid automaton model checking approach to the hybrid system.

## INTRODUCTION

Currently most of vehicle innovations are mainly driven by electronics. In modern passenger cars there are up to 60 computing units (e.g. Volkswagen Phaeton). Automotive software runs on so-called Electronic Control Units (ECU). An ECU consists of a microcontroller and memory, next to power electronics to drive sensors and actuators. The software implements control algorithms, combines the sensor values and calculates some meaningful actuator signals. In a study between automotive software developers it was found that in recent years cost of software development rose and exceeded one third of total vehicle production cost (1).

[i] This work was supported, in part, by Science Foundation Ireland grant 03/CE2/I303_1 to Lero - the Irish Software Engineering Research Centre

Among the many factors driving this rise in development costs is the increasing importance for the electronic systems controlling vehicle parts to meet requirements of safety, fuel economy, environmental requirements, comfort and convenience, multimedia or entertainment services. Moreover a trend can be seen where more functionality will be put into software to allow reduction of hardware sensor cost (2). To help progress of electronics in automotives many research projects were established. The most prominent dealing with automotive systems, especially software in vehicles is named AUTOSAR (3). In short it is focused on software architecture in the ECU.

## HYBRID SYSTEM DEFINITION

However, as mentioned, each ECU is a part of a more complex system, dealing with mechanical and electrical properties of the vehicle. From an electronic point of view this complex system can be seen as a hybrid system. In this context the hybrid system is understood as a system whose behaviour can be described by two complementary parts, continuous and discrete dynamics. In details, the hybrid system has a continuous evolution and occasional jumps. If the system behaviour is described as an automaton, which is most common, then the jumps represent the changes of automaton state where transition is the response to external events or to the continuous events. A continuous evolution is associated to each automaton state by ordinary differential equations. Each state may have different initial conditions and the structure of the equations. While this informal definition may be understood as simple, the precise definition of the evolution of the system is more complex. Moreover the complexity of those hybrid systems is increasing with more detailed design, which brings a challenge for analysis and verification of them from both, the theoretical point of view and the implementation side.

## SIMULATION OF HYBRID SYSTEM

For the historical reasons the first computer tool to design complex systems is a computer based simulation. The aim of simulations is to avoid extensive testing after manufacture and therefore reduce the cost and time associated with it. One of the most popular tools for any numerical computation for physical or mathematical problems is Matlab with Simulink (4). The other tool gaining popularity is the MODELICA object oriented language (5). Unfortunately the degree of confidence in the correctness of the design is limited as unpredicted interactions with the environment go unchecked since the input data size is too large to allow full analysis. In addition, building prototypes of the system suffers identical problems with the rise of the system complexity. Because of that formally verifying high level design of hybrid system can be very useful for safety critical systems like most automotive-related systems. By building a mathematical model of the system it is possible to use automated model checking methods to prove that all requirements are met or all possible system input sequences are checked. This is compared to simulation where only part of the all possible system inputs will be checked.

## VERIFICATION OF HYBRID SYSTEM

Formal verification may be seen as a very interesting concept, helping with designing hybrid systems, because it avoids the main drawback of simulation, a lack of guarantee for design correctness. Formal verification is intended to prove that some properties hold all the interesting modes of operation of the system which is being analysed. However its usefulness is limited by the complexity of the analysis, which might be very large when the size of the

designed system increases. This can be overcome by a design methodology which helps to tackle the complexity of the design and may help in finding a simpler and cleaner solution. Formal verification algorithms determine whether a mathematical model of the system meets a specification of requirements given in temporal logic. With discrete systems with a finite number of states, model checking can be used in validation of hardware or communication protocols. It is also useful in checking real time systems, which have discrete time (6). Ideally the results are obtained either as an exact or a conservative over-approximation of the behaviour of the system, particularly as the set of reachable states. An exact computation is possible with linear hybrid automata, which are defined by linear predicates and piecewise constant bounds on the derivatives.
Hybrid systems can be verified by the model checking algorithm of the hybrid automaton. It analyses the defined properties to determine if they are violated in any state reachable by the automaton (6). A hybrid automaton itself can be described as an abstraction of a finite state machine, which allows continuous variables. The discrete actions are modelled by moving through a finite set of control locations. In addition the continuous actions are modelled by real variables with values changing continuously over time, according to the ordinary differential equations describing them (7).
Another advantage is that the hybrid automata can be designed in parallel configuration, which is useful for describing large, complex systems. Each part of the system can be described by a separate hybrid automation with the possibility of communicates between automata (8).

## HYBRID SYSTEM DESIGN

Most wide spread approaches to hybrid system design are based on the bottom-up design methodology. This begins with information gathered as a result of system description by physical rules and equations. Objects in the system are related to physical objects of the system and interactions between them are made according to the physical equations (9). This bottom-up methodology is focused on detailed description and its relevance to the physical world. Equations describing the system can complicate design, bringing too many detailed equations into the initial approach. This is clearly visible with large systems described by many physical laws. There is no idea of system abstraction different to using less realistic physical equations. Moreover, there is no established rule for keeping track of the changes. Every case is explored on its own with different levels of abstraction and its solution is found by numerous experiments (9).
On the other hand, the usefulness of formal methodology is limited by lack of a well defined methodology which would allow coping with it. This is also a case when designing hybrid systems with the help of hybrid automata. Moreover, there are no tools for interactive model building and analysis interpretation when too much complexity can only be beaten by using appropriate abstraction of detailed models. In addition, aids should be given to translate informal requirements specifications into formal specifications, since formal specifications are quite difficult to write for practical engineering.

## AUTOMOTIVE SYSTEM DESIGN

Compared to above design approaches the automotive software and system design is more sophisticated. In the automotive industry model based specification techniques are becoming more and more popular. This allows for complete, consistent, and explicit specification of software and hardware for car specific networks of control units. Partially it comes from mechanical design, where design models are well established. This relies on a top down

approach with stress on recurrence in finding key components of the system and partitioning them in to subsystems (10).
In this environment, model based approaches provide methodology support to manage the integration of logical functions as interactive components within distributed networks of ECU's.
In addition well defined models are a source for analysis, validation, and verification activities. This kind of analysis is listed by the abstraction level of the system. Categorization is based on flow and abstraction of information related to the design of a system model.
The process reflecting this partitioning is standardised as a V-model approach and seen as one consisting of three parts (11). It is a well defined process, following the top down approach to design. It starts with a top level description of the system process and moves on step by step from this view into more detail. With iteration of the method the system is decomposed into smaller modules on a lower level. Top down view requires identifying at least the major highest level system requirements and functions and then breaks them down in iterative steps until function specific modules can be designed. This process on each level brings a more detailed description and continues until atomic level of problem decomposition is achieved.
This focus into the details can be seen in changing scope from system engineering to software engineering with implementation of code (12). It might be hard to design a correct and more detailed system, without considering interactions with its environment. The simulation might overcome this problem, but leaves design with uncertainty for its correctness. Moreover, if the modules building automotive system come from different OEMs the simulation might not be sufficient for system validation and formal methods are needed.
The proposal of this work is that hybrid system design methodology is focused on both, the top layers named system engineering and software system engineering. Only software design and implementation is not considered over here. A software process model like the V-model requires also an environment model. In most cases this model is limited to a small subset of the whole system model, a diagram or a class. However design methodology of hybrid systems requires an environment model for a more formal approach to the design of an embedded system, at least for formal verification or testing.

## NEW DESIGN METHODOLOGY

The proposed design methodology will be introduced by describing its usage on the example of developing Electronically Controlled Suspension (ECS) system for a vehicle (13). The following example will show research done on the methodology design. During description of the methodology it is assumed that design flow is supported by a tool, currently in development.
The suspension of the vehicle has developed over the years to a high level of complexity and sophistication. In the past car makers used metal spring elements, but after years have switched to hydro-pneumatic or pneumatic elements to isolate the vehicle for road irregularities (14). The nature of the active suspension system is hybrid. It needs to constantly adapt digital control strategy to changes of analogue environment. The control input depends on the discrete state where the system is at, and influences the continuous state of the system in parallel. Moreover it in turn, determines the transition between discrete states (15).
This can be transformed into a list of requirements for the system. Starting from information about vehicle dynamics and leading to detailed description of the ECS task.
Gathering of the detailed requirements can be done by developing use case studies. Use cases can help see requirements from different perspectives. All of them can be linked in one (see Figure 1), where the actor named vehicle wheel dynamics represents the behaviour of the wheel and by that is a part of the suspension system. In addition, the actor named vehicle

wheel pneumatics represents the active parts of the vehicle pneumatics system. The actions which they can take and, related to the suspension, are changing the vehicle level and set the vehicle level respectively.
The basic goal or the main requirement of the system is to maintain the vehicle chassis of the same level. The change in vehicle level caused by the vehicle dynamics actor should trigger the action of the actor, vehicle pneumatics.
On this, high abstraction level, the value of variable describing the difference between changed vehicle level and desired vehicle level is not calculated. There are two possibilities the vehicle level set up by the actor vehicle dynamics can be higher or lower compare to the desired vehicle level. The actor named vehicle pneumatic system should behaviour respectively to that; it has to minimize that difference. There is a need for constant recalculation of the difference between desire and actual vehicle level.

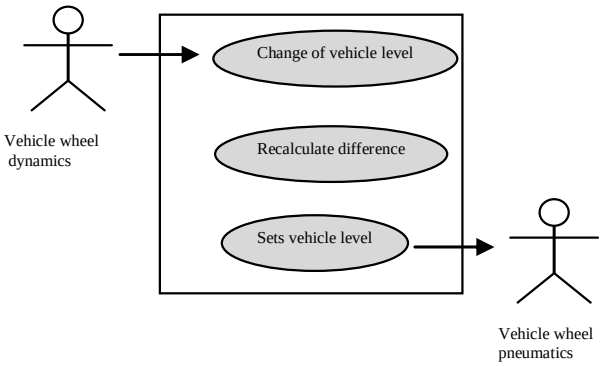


Figure 1: Vehicle active suspension use case diagram

This approach describes behaviour of the active suspension system and basic services that ECS delivers. It could be routed from the source of information to its sink to observe and record changes in vehicle levels caused by different behaviours of the vehicle on the road, see figure 2 for representation. This record change should be used for recalculation of the difference between given and desired vehicle level. The difference should be used as set up information for actuators responsible for vehicle level.

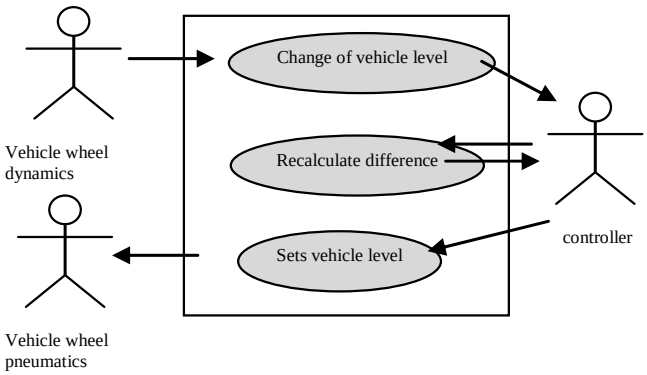


Figure 2 Active suspension controller use case diagram

A different perspective of the flow of information or data between actors describes in a consistent scenario, where vehicle wheel dynamics actor is the main source of data. This linear scenario is a scenario with no branches in it. Any variation in the scenario should lead to other scenarios having different prerequisites. The only exception not considered in this example is error handling, to keep the example as understandable as possible. In relation to control engineering this approach can be seen as an example with only one main control loop

and with all possible errors taken in to account during design. This preparation of error handling allows dealing with this at a later, more detailed, stage of the design. The possibility of such branches is highlighted by verification during the final stage.

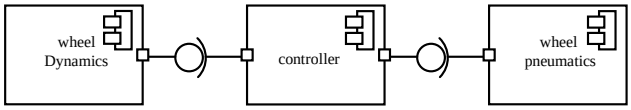

Figure 3 Basic suspension partitioning

During analysis of use case graphs those actors can be described as data stores similar to those on data flow diagrams. Each of the graphic symbols has its representation in formal logic language to allow transformation from use case like diagram (see figure 2) to the other one described in figure 3. The actors are crucial for the finding and definition of main system components, later related as a main system objects, like on figure 3.
The starting point of analysis of vehicle pneumatic suspension system is shown in figure 4, where the compressor actor or release valve actor can interact with the pneumatic actuator. The transfer of this description to the data store can be seen in figure 3. Each of those elements has a possibility of embedding a subsystem describing its behaviour in more detail, figure 4. Moreover each of them has to track information needed for the hybrid automata design of the lower stage. The embedding of hybrid automata as a part of the other automata will be unfolded for the verification tool.
During verification each automata must be explicitly design not to be embedded within another. However in earlier stages of the process it can be hidden for easier design. As a verification tool in this methodology HyTech (16) will be used. It is the most complete tool based on hybrid automata and the only one that allows a parametric approach (17). Moreover, HyTech is better suited to high level system description, where the continuous variables are either simple dynamics or it is possible to exchange them for the one with simple dynamics. This would be an advantage in the top down design approach.
Common design problems have the hybrid system used instead of control systems. The basic partitioning of the system in the proposed methodology can be related to partitioning of a system from the classical control loop theory.

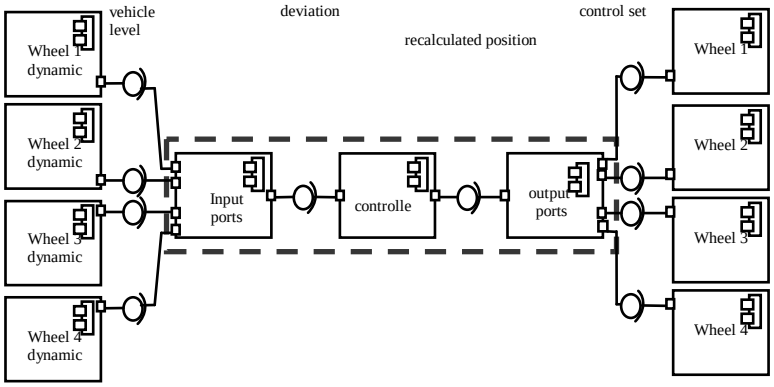

Figure 4 Controller hierarchy

This example and most of the hybrid systems considered in design can be defined as a system which constantly tries to adjust itself. The aim of this methodology is to ensure that self adjustments will be always performed in the safety zone of the system. The ECS tries to adjust the position of vehicle chassis with disturbances (errors) coming from the road. The classical control system with feedback consists of a controller performing adjustment calculations, sensors observing the disturbance in the system’s environment and the

environment itself. These are named a control round or loop and will be used in this methodology for more detailed design. The basic elements of this control loop are taken from partitioning the system into objects (see figure 3). It is done to allow express information transferred between objects in time (see figure 5).
The control information exchanged between objects can be referred to the role of the each object in the system. In the example the wheel dynamics object provides information about the current vehicle level. The value of the wheel dynamics object should be used to set up new vehicle level. In this case the absolute value of the vehicle is not important for the system. Only the change of the level or the level deviation is important. The relation of changes to distance covered by the vehicle is out of the scope, while the vehicle level changes over time are relevant. In addition, not every change of level should be taken into account; some of them come from fluctuation connected with vehicle dynamics.

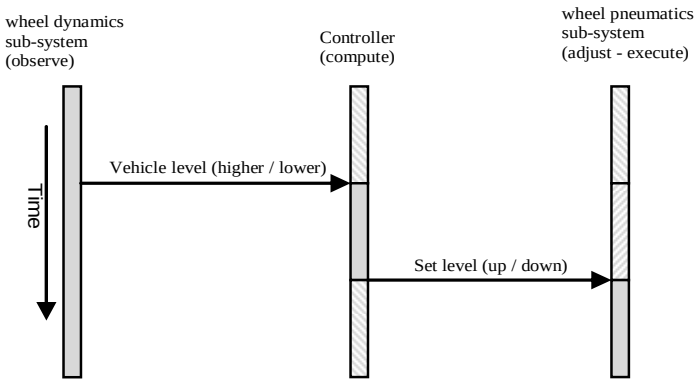

Figure 5 Basic controls round

Another aspect is the delay in time after which vehicle systems will react to changes of the vehicle level. It introduces the problem of filtering some of the vehicle level changes. Changes might be too small, or too fast and the system will not be able to react. That can be other parameters to verify and the process can bring the information what are safe values for each of them. The analysis phase ends with overall structure of the system as a group of defined object and links between them, as seen on figure 6.

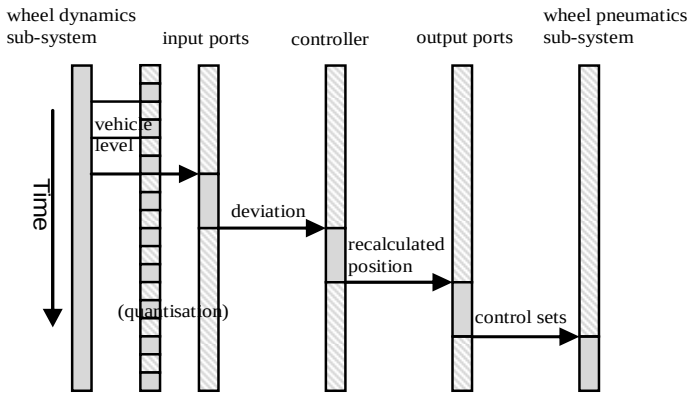

Figure 6 Control data flow

Moreover objects are linked by shared interfaces which are used to pass state of variables between objects as information about state of the system (see figure 4). In different design example this may be used to coordinate more than one hybrid automata. The model of the system should not only consist the control software but also models of it environment. It is

necessary for the verification of the system and parameters finding. Labels given to information transferred between object of the system, (see figure 6) are used as a hybrid automata edges of the controller, as shown on figure 7.

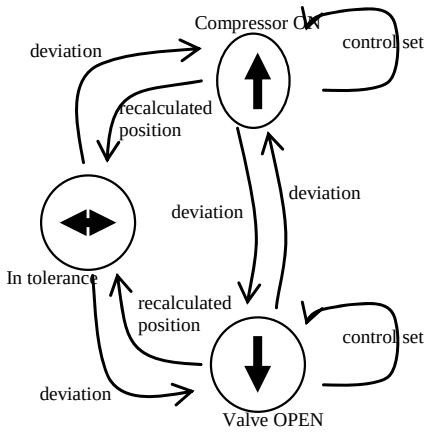


Figure 7 Controller

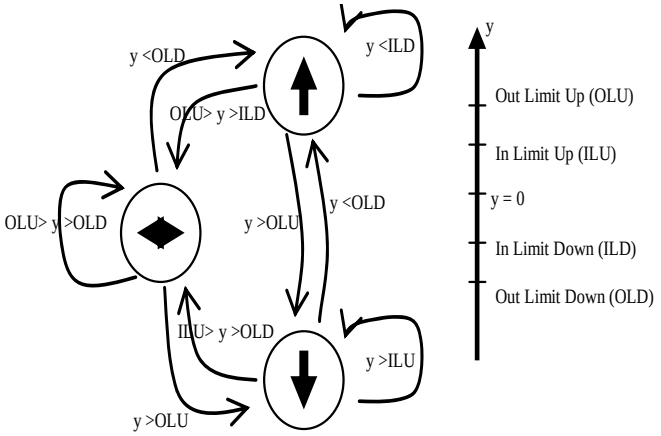


Figure 8 Controller with parameters (y is a vehicle level in ECS)

This description is a starting point of transforming fixed controller example to the parameterized one, figure 8. Parameter analysis of the hybrid system gives a possibility for fining optimal configuration of variables describing the system and by that better performance of the system. Final description of the parameters must be specified in requirements. Results are also validated if they pass safety requirements of the system. This design is final design of controller automata, from which the code for the verification tool will be generated.

A critical part of the development is the formulation of guidelines for discretization process, the transfer from continuous time to discrete time description. It is shifted in design to the latest possible moment, not to influence the design. The resolution of the system can be a design parameter as well; it has direct impact on the discretization. This parameter considered during automotive system design, has direct relation to the cost of the system. The figure 9 shows ECS controller with parameters where y is a current vehicle level and its desirable value is zero. Any disturbance of this value is measured against parameters, limits in which system can change itself.

## METHODOLOGY EVALUATION

As was mentioned before the methodology is supported by the tools currently in the development. The graphical language used in methodology can be seen as a domain specific language build on the top of UML. As an example the UML and its extensions are also used in the AUTOSAR project to model various parameters of automotive systems (18).
The other project related to the design of automotive hybrid systems in formal way is AutoMoDE (19), where the code generation ASCET tools were combined with the research outcome of the AutoFocus (20) project. Proposed over there approach is based on the HyCharts (21), which tries to extend state charts for the continuous domain. This problem was overcome by partitioning the system to the continuous and discrete space in the earliest stage of the development. It is not different form other existing approaches, like mentioned before Matlab. Moreover research around the Matlab tries to extend its usability by verification with use of formal methods (22). Unfortunately, all that approaches relay on early partitioning in to

discrete and continuous domain. This may be still appropriate for detailed design of the specific part of the system.
However as it can be seen from broader perspective currently hybrid system design methodologies are focused on the separate design of each of the system objects. It can be described as a bottom up approach. This design is focused on correctness or finding significant parameters of each object on its own. Moreover approach is used also for verification of the system, where is assumed that each formally correct object is connected to other verified objects and by that whole system is verified. That may not be a case for the complex system.
The new and different approach, proposed in this work, is based a top down design inherited from automotive system design approach. It is focused on a whole system and its role. It would tackle the complexity on a higher level with the possibility to consider more than one structure of designed system. Moreover it would allow verifying the correctness of whole system.
The problem of composition of each verified object in the system is explored in many works (23), but it also may lead to a problem where an object verified with a set of parameters for one task, even an obvious one, may not be the best choice for other task. In high level description of a system parameters or symbolic constants are often used with not specified real values. In most cases those parameters would gain values later in the design process during implementation. The parametric analysis of the system would determine necessary and sufficient constraints for parameters of the system to operate safety. Computing limitations in which system will not violate safety requirements would help define the optimal set of parameters for the system work. The case study used as an example was also described by different approaches (24). However over there the focus was on the verification itself without any information how to handle the complexity of the system in design. The other assumption over there was that the designer is familiar with formal methods as a verification tool. This may not be a true for every system engineer. Proposed in this work design methodology allows the system engineers to have benefits of formal methods by using more understandable and valid for them approach.
By verifying the model of a system with the model checker the system designer has a formally proven model system which can be use for the actual implementation in the ECU. By that the only thing to be verified is actual implementation and it correctness against the model.

REFERENCES


(1) M. Broy, I.H. Krueger, A. Pretschner, C. Salzmann; Engineering automotive software; Proceedings of the IEEE; 2007;
(2) K. Haenninen, J. Maeki-Turja, M. Nolin; Present and future requirements in developing industrial embedded real time systems – interviews with designers in the vehicle domain; Engineering of computer based systems; IEEE; 2006
(3) H. Heinecke et all; AUTOSAR Current results and preparations for exploration; Software in the Vehicle; 2006
(4) Stuermer I.; Travkin D.; Automated Transformation of MATLAB, Simulink and Stateflow Models; OMER – Object oriented Modelling of Embedded Real Time Systems Proceedings; 2007;
(5) T.A. Johnson, C.J. Paredis, J.M. Jobe, R.Burkhart; Modelling continuous system dynamics in SysML; IMEC; 2007
(6) R. Alur, T. Henzinger P. Ho; Automatic Symbolic Verification of Embedded Systems; IEEE Transactions on Software Engineering; vol. 22; 1996

(7) T. A. Henzinger, The Theory of Hybrid Automata, Proc. Of the 11th Annual IEEE Symposium on Logic in Computer Science, p. 278-292, 1996
(8) T. A. Henzinger, P. H. Ho, H. Wong-Toi, A User Guide to HyTech, Tools and algorithms for Construction and Analysis of Systems, p. 41-71, 1995
(9) T.A. Henzinger, W. Wong-Toi; Using HYTECH to Synthesize Control Parameters for a Steam Boiler; Lecture notes in computer science; issue 1165, Springer Verlag; 1996
(10) M. Toengren, D. Chen, I. Crnkovic; Component based vs. model based development a comparison in the context of vehicular embedded systems; Sofware Engineering and Advanced Applications; IEEE; 2005
(11) chuppan V., Russwurm W.; A CMM Based Evaluation of The V-model 97; Lecture notes in computer science; issue 1700; Springer Verlag; 2000
(12) ettenbach C. W.; Analyse des V-Modells als Entwiclungsstandard fuer IT-Systeme des Bundes; FernUniversitaert in Hagen; 1998
(13) Bosch Automotive Handbook; Robert Bosch GmbH; 2005
(14) J. Darling, R.E. Dorey, T.J. Ross-Marlin; A low cost active anti-roll suspension for passenger cars; Journal of Dynamics Systems, Measurement and Control; 1992
(15) N. Elia, B. Brandin; Verification of an Automotive Active Leveller; Proceedings of American Control Conference; IEEE; 1999
(16) T. A. Henziger, P. H. Ho, H. Wong-Toi, HyTech: A Model Checker for Hybrid Systems, Int. Journal on Software Tools for Technology Transfer, vol. 1, p. 110-122, 1997
(17) B. Tavernier, Calife: A Generic Graphical User Interface for Automata Tools, Electronic Notes in Theoretical Computer Science, vol. 110, p. 169-172, 2004
(18) P. Cuenot, P. Frey, R. Johansson, H. Loenn, M.-O. Reiser, D. Servat, R. Tavakoli Kolagari, D.J. Chen; Developing automotive products using the EAST-ADL2, an AUTOSAR compliant architecture description language; Proc. of ERTS 2008, Toulouse, France.
(19) A. Bauer et all; AutoMoDe – Notations, Methods, and Tools for Model Based Development of Automotive Software; SAE 2005
(20) F. Huber, B. Schaetz, G. Einert; Consistent Graphical Specification of Distributed Systems; Lecture notes in computer science; issue 1313; Springer Verlag; 1997
(21) R. Grosu, T. Stauner; Modular and Visual Specification of Hybrid Systems: An Introduction to HyCharts; Formal methods in system design; Vol. 21; Springer; 2002
(22) A. Cavalcanti, P. Clayton; Verification of Control Systems using Circus; Proceding of IEEE International Conference on Engineering of Complex Computer Systems; 2006
(23) D. Sinclair; Using an Object oriented methodology to bring a hybrid system form initial concept to formal definition; Lecture notes in computer science; issue 1201; Springer Verlag; 1997
(24) N. Elia, B. Brandin; Verification of an Automotive Active Leveller; Proceedings of American Control Conference; IEEE; 1999